

Corso di Laurea in Ingegneria Informatica



Corso di Reti di Calcolatori I

Roberto Canonico (roberto.canonico@unina.it)

Giorgio Ventre (giorgio.ventre@unina.it)

**Il livello trasporto:
tecniche di trasmissione affidabile dei dati**

**I lucidi presentati al corso sono uno strumento didattico
che NON sostituisce i testi indicati nel programma del corso**

Nota di copyright per le slide COMICS



Nota di Copyright

Questo insieme di trasparenze è stato ideato e realizzato dai ricercatori del Gruppo di Ricerca COMICS del Dipartimento di Informatica e Sistemistica dell'Università di Napoli Federico II. Esse possono essere impiegate liberamente per fini didattici esclusivamente senza fini di lucro, a meno di un esplicito consenso scritto degli Autori. Nell'uso dovranno essere esplicitamente riportati la fonte e gli Autori. Gli Autori non sono responsabili per eventuali imprecisioni contenute in tali trasparenze né per eventuali problemi, danni o malfunzionamenti derivanti dal loro uso o applicazione.

Autori:

Simon Pietro Romano, Antonio Pescapè, Stefano Avallone,
Marcello Esposito, Roberto Canonico, Giorgio Ventre

Realizzare una trasmissione affidabile



- Se il livello rete è inaffidabile:
 - Presenza di errori
 - Perdita di pacchetti
 - Ordine dei pacchetti non garantito
 - Duplicazione di pacchetti
 - Inoltre bisogna tenere in considerazione:
 - Le risorse del computer ricevente:
 - Controllo di flusso
 - Le risorse della rete
 - Controllo di congestione

3

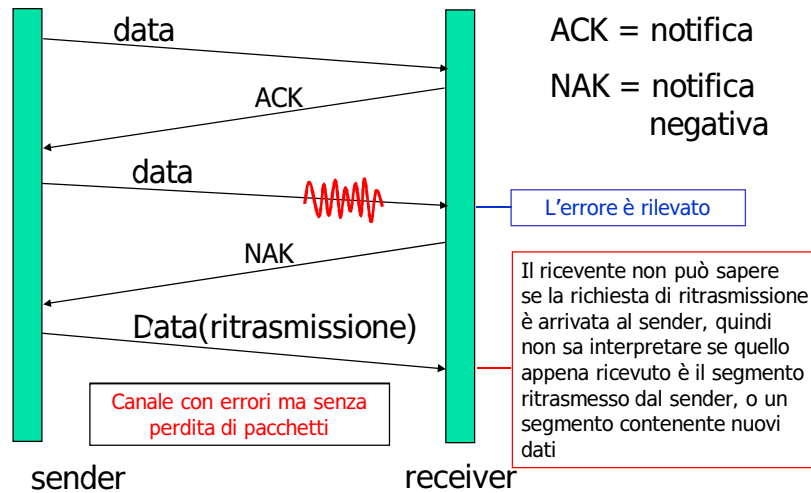
Realizzare una trasmissione affidabile - 2



- **Soluzioni:**
- Rete che presenta errori di trasmissione:
 - In questo caso il ricevente deve effettuare:
 - Rilevamento degli errori e:
 - 1) Correzione degli errori
 - oppure
 - 2) Notifica al mittente
 - Richiesta ritrasmissione
 - La prima soluzione introduce complicazioni, la seconda introduce possibili duplicazioni sulla rete che il ricevente non è in grado di interpretare

4

Realizzare una trasmissione affidabile - 3



5

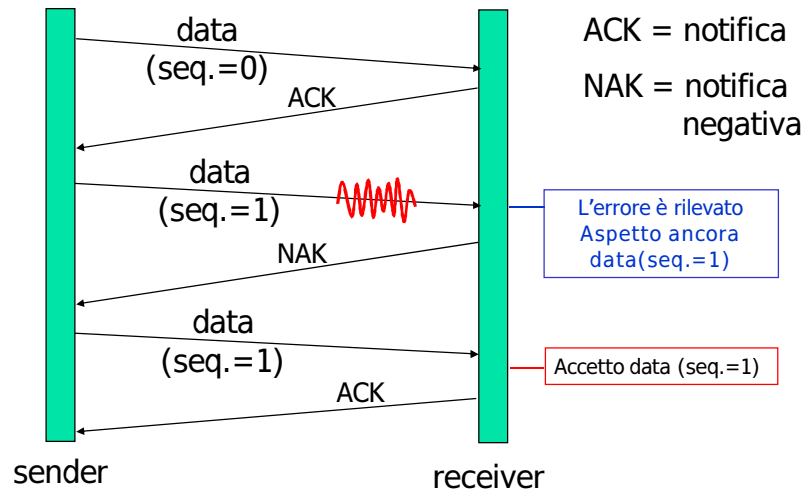
Realizzare una trasmissione affidabile - 4



- Per risolvere il problema dei duplicati che il ricevente non è in grado di interpretare, occorre inserire nell'header del segmento da inviare un'ulteriore informazione:
 - **numero di sequenza**
- Nel caso di protocolli che inviano un messaggio e quindi aspettano un riscontro prima di ritrasmettere un nuovo messaggio (**stop & wait**), è sufficiente un numero di sequenza su un bit (0,1). Vediamo un esempio:

6

Realizzare una trasmissione affidabile - 5



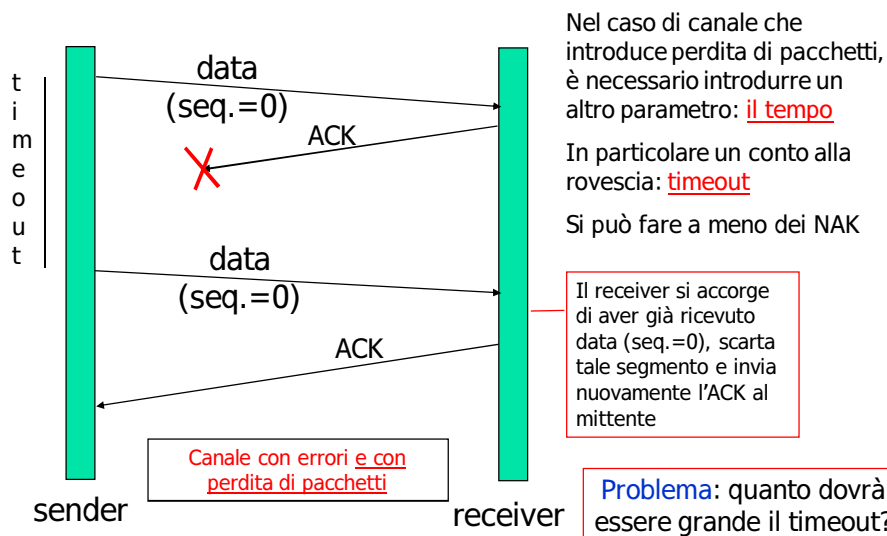
7

Un protocollo senza NAK



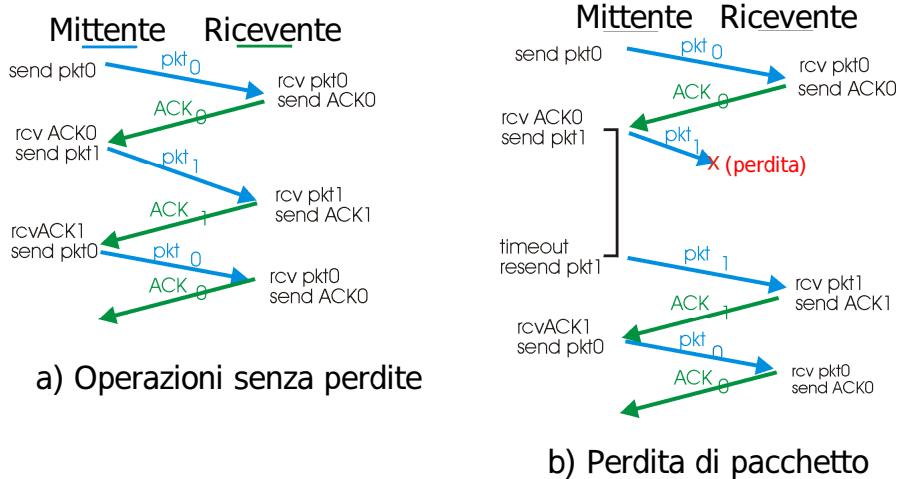
- Stessa funzionalità del precedente, utilizzando soltanto gli ACK
- Al posto del NAK, il destinatario invia un ACK per l'ultimo pacchetto ricevuto correttamente
 - il destinatario deve includere *esplicitamente* il numero di sequenza del pacchetto con l'ACK
- Un ACK duplicato presso il mittente determina la stessa azione del NAK: *ritrasmettere il pacchetto corrente*

Realizzare una trasmissione affidabile - 6

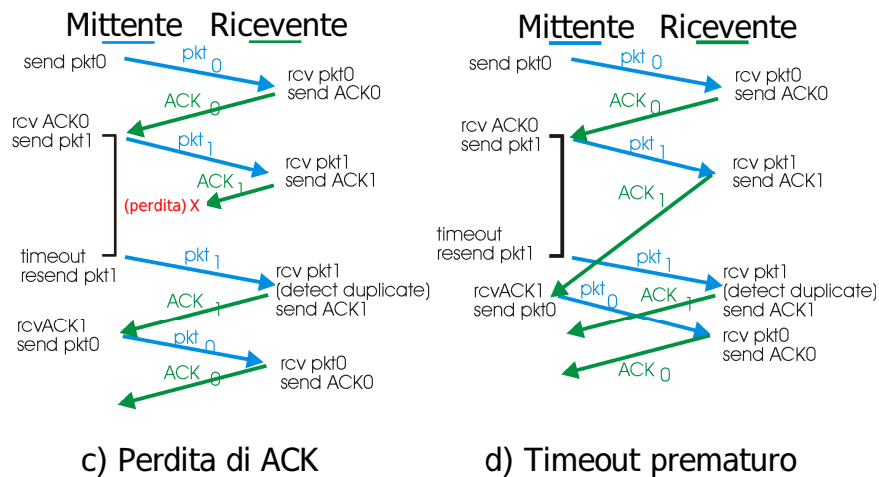


9

Stop&Wait: Ricapitolando (1/2)



Stop&Wait: Ricapitolando (2/2)



... anche detto "protocollo ad alternanza di bit"

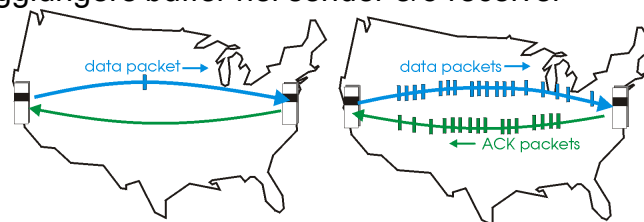
Aumentare l'efficienza



In alternativa al semplice Stop & Wait ...

Pipelining: il mittente invia pacchetti prima di ricevere il riscontro dei precedenti

- Occorre aumentare l'intervallo dei num. sequenza
- Aggiungere buffer nel sender e/o receiver



(a) a stop-and-wait protocol in operation

(b) a pipelined protocol in operation

- Due alternative per il pipelining: **go-Back-N**, **selective repeat**

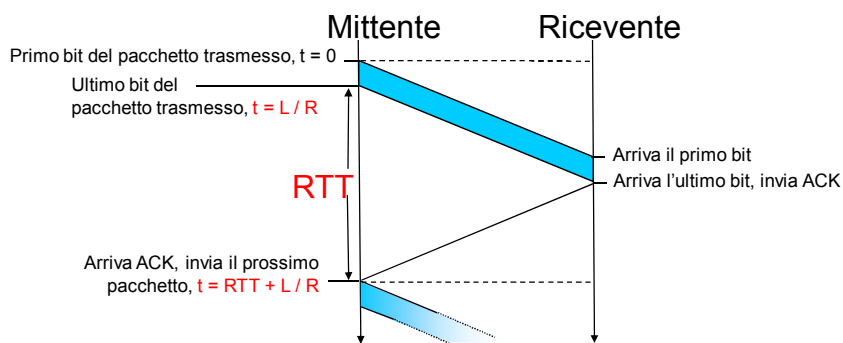
Performance



- Funziona, ma le performance...
- Esempio: 1 Gbps link, 15 ms e-e prop. delay, 1KB packet:

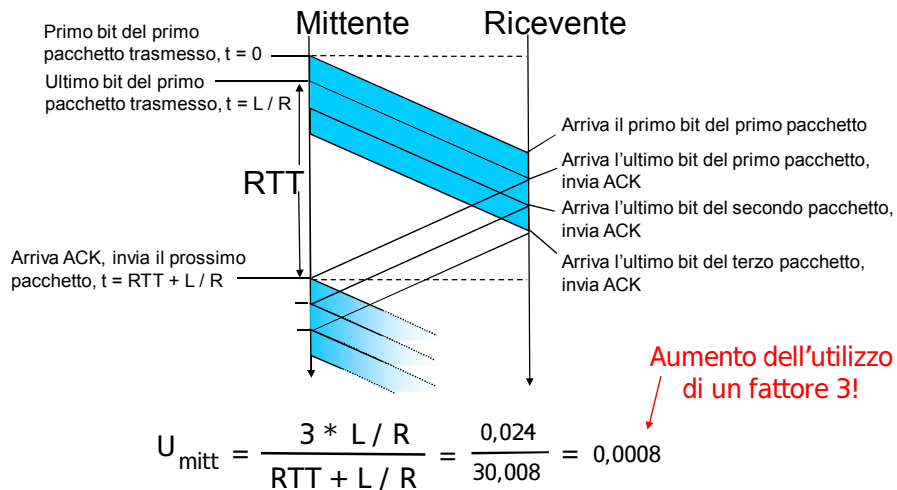
$$T_{\text{transmit}} = \frac{L \text{ (packet length in bits)}}{R \text{ (transmission rate, bps)}} = \frac{8\text{kb/pkt}}{10^{**}9 \text{ b/sec}} = 8 \text{ microsec}$$

Funzionamento con stop-and-wait



$$U_{\text{mitt}} = \frac{L/R}{RTT + L/R} = \frac{0,008}{30,008} = 0,00027 = 0,027 \%$$

Pipelining: aumento dell'utilizzo

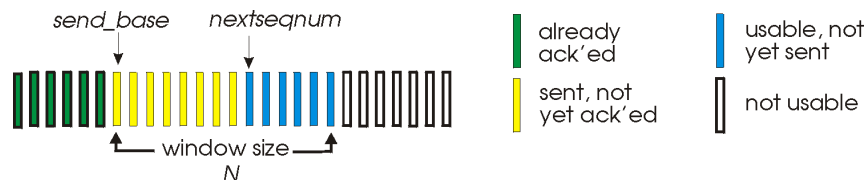


Go Back-N



Sender:

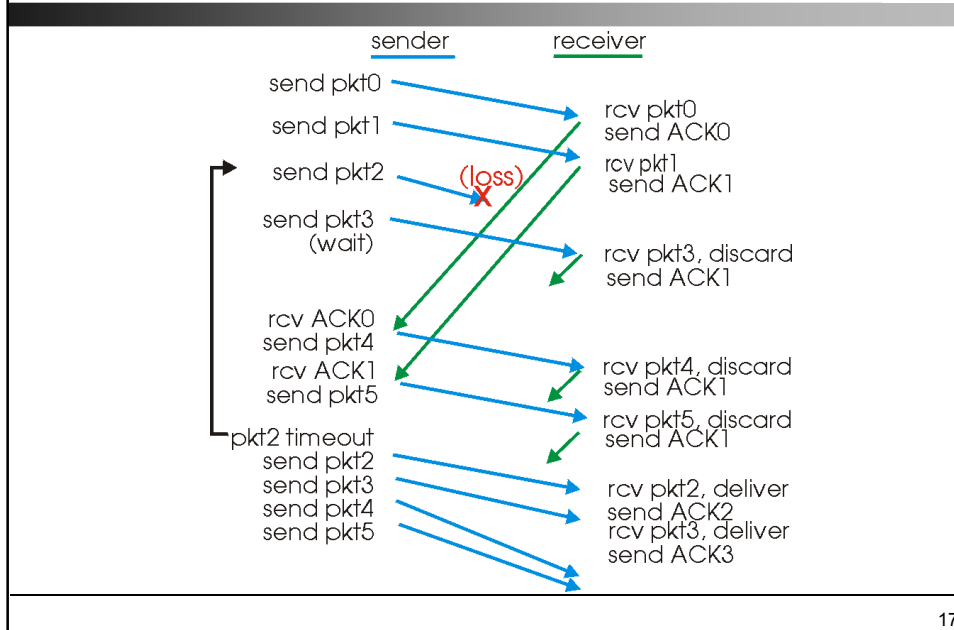
- Nell'header del segmento k-bit per il num. sequenza
- Una finestra di max N pacchetti senza riscontro
- ACK numerati



- ACK cumulativo: ricevere ACK(n) significa che tutti i pkts precedenti l'n-esimo sono stati ricevuti correttamente
- Un timer per ogni pacchetto trasmesso e non riscontrato
- *timeout(n)*: ritrasmetti pkt n e tutti i pacchetti che seguono n

16

Go Back-N in azione



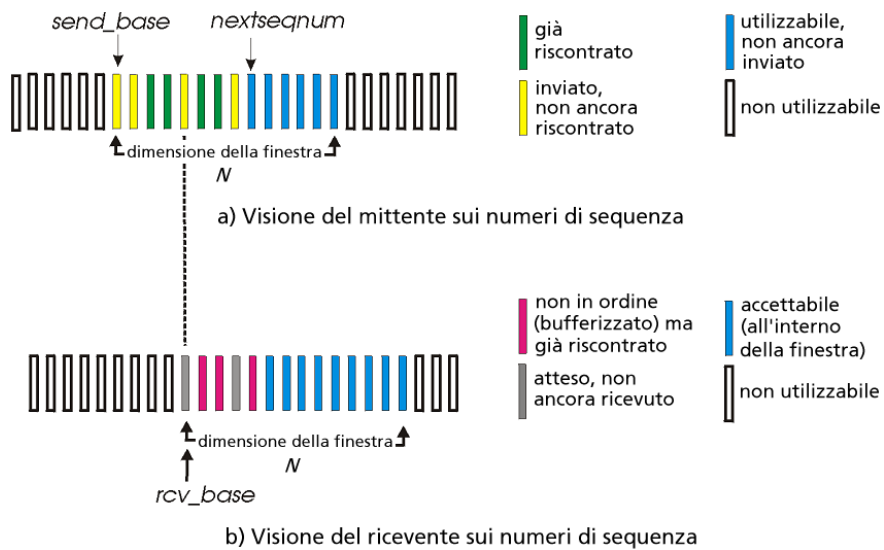
17

Ripetizione selettiva

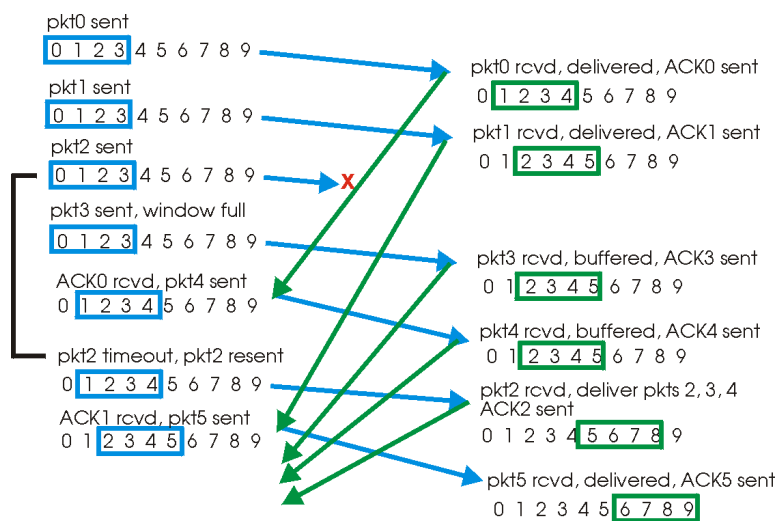


- Il ricevente invia riscontri *specifici* per tutti i pacchetti ricevuti correttamente
 - buffer dei pacchetti, se necessario, per eventuali consegne in sequenza al livello superiore
- Il mittente ritrasmette soltanto i pacchetti per i quali non ha ricevuto un ACK
 - timer del mittente per ogni pacchetto non riscontrato
- Finestra del mittente
 - N numeri di sequenza consecutivi
 - limita ancora i numeri di sequenza dei pacchetti inviati non riscontrati

Selective Repeat



Selective Repeat in azione



Ripetizione selettiva: dilemma



Esempio:

- Numeri di sequenza: 0, 1, 2, 3
- Dimensione della finestra = 3

- Il ricevente non vede alcuna differenza fra i due scenari!
- Passa erroneamente i dati duplicati come nuovi in (a)

D: Qual è la relazione fra lo spazio dei numeri di sequenza e la dimensione della finestra?

